Exhibit 1

Appl. No.: 10/002,461
Applicant(s): Keith R. Slavin
Filed: November 1, 2001
Title: LOW POWER, HASH-CONTENT ADDRESSABLE MEMORY ARCHITECTURE
Art Unit: 2189
Examiner: Reba I. Elmore

Docket No.: DB000955-000

DECLARATION UNDER RULE 131

I, Russell D. Slifer, state as follows:

1.   I am Chief Patent Counsel for and authorized representative of Micron Technology, Inc., the assignee of U.S. patent application serial no. 10/002,461 ("the '461 application").

2.   In connection with the '461 application, an Invention Disclosure form identifying Keith R. Slavin as the inventor was submitted to the Micron patent department. A copy of the first and last pages of the Invention Disclosure form are attached as Exhibits A and B, respectively. The last page of the Invention Disclosure form is dated June 27, 2001.

3.   Attached to the Invention Disclosure form were four figures, attached hereto as Exhibit C-1/4 through Exhibit C-4/4. Each of the figures of Exhibit C was read and understood by a witness, who signed and dated each of the figures on June 27, 2001.

4.   Also attached to the Invention Disclosure form was a multi-page disclosure document detailing the construction and operation of the invention. Eight pages from that multi-page disclosure document are attached hereto as Exhibit D-1/8 through Exhibit D- 8/8. Each page of the multi-page disclosure document was read and understood by a witness, who signed and dated each of the pages on June 27, 2001.

5.   I have reviewed and am familiar with the '461 application.
   • The Figure on Sheet 1/4 of Exhibit C shows the same subject matter as Figure 1 of the '461 application.

- The Figure on Sheet 2/4 of Exhibit C shows the same subject matter as Figure 2 of the '461 application.

- The Figures on Sheets 3/4 and 4/4 of Exhibit C show the same subject matter as Figure 3 of the '461 application.

6. For each of the method limitations in claim 1, there is a corresponding hardware element shown in Figure 1 of the '461 application which provides that function. The correspondence is as follows:

inputting an input word to a plurality of hash circuits, each hash circuit being responsive to a different portion of said input word [see input lines to elements 24(1) – 24(n)];
outputting a hash signal from each hash circuit [see output lines from 24(1) – 24(n)];
enabling portions of a CAM in response to said hash signals [see enable lines output from element 26];
inputting said input word to said CAM [see input line to element 20 from element 28];
comparing said input word in the enabled portions of said CAM [function of element 20]; and
outputting information responsive to said comparing [output from element 20].

7. Because the same hardware and connections that are shown in Figure 1 of the '461 application are also shown in the Figure on Sheet 1/4 of Exhibit C, the invention disclosure dated June 27, 2001 supports claim 1 in the same manner that Figure 1 of the '461 application supports claim 1.

8. Claim 1 is also supported by the paragraph labeled "Summary" on page D – 1/8 and by the details of the invention disclosure as follows:

inputting an input word to a plurality of hash circuits, each hash circuit being responsive to a different portion of said input word [see paragraph labeled "Hardware Parallelism using Hashing Circuits and Matching in TCAM" on page D-3/8 and the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
outputting a hash signal from each hash circuit [see first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
enabling portions of a CAM in response to said hash signals [see third paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
inputting said input word to said CAM [see first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
comparing said input word in the enabled portions of said CAM [see paragraph labeled "TCAM Matching" on page D-4/8 and third paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8]; and

- 2 -

outputting information responsive to said comparing [see the paragraph entitled "TCAM Output" on the bottom of page D-4/8 and top of page D-5/8].

9.   For each of the method limitations in claim 8, there is a corresponding hardware element shown in Figure 1 of the '461 application which provides that function. The correspondence is as follows:

hashing a comparand word [see elements 24(1) – 24(n)];
precharging certain portions of a CAM in response to said hashing [see enable lines output from element 26]; and
inputting said comparand word to said CAM [see input line to element 20 from element 28].

10.   Because the same hardware and connections that are shown in Figure 1 of the '461 application are also shown in the Figure on Sheet 1/4 of Exhibit C, the invention disclosure dated June 27, 2001 supports claim 8 in the same manner that Figure 1 of the '461 application supports claim 8.

11.   Claim 8 is also supported by the details of the invention disclosure as follows:

hashing a comparand word [see paragraph labeled "Hardware Parallelism using Hashing Circuits and Matching in TCAM" on page D-3/8 and the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
precharging certain portions of a CAM in response to said hashing [see paragraph labeled "TCAM Match Line Power Dissipation" on page D-5/8]; and
inputting said comparand word to said CAM [see first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8].

12.   For each of the method limitations in claim 15, there is a corresponding hardware element shown in Figure 1 of the '461 application which provides that function. The correspondence is as follows:

inputting an Internet address to a plurality of hash circuits, each hash circuit being responsive to a different portion of said address [see input lines to elements 24(1) – 24(n)];
outputting a hash signal from each hash circuit [see output lines from 24(1) – 24(n)];
using said hash signals to identify portions of a CAM [function of element 42 of Figure 2 and element 26 of Figure 1];
inputting said address to said CAM [see input line to element 20 from element 28];
comparing said address in only the identified portions of said CAM [function of element 20]; and
outputting port information in response to a match being found in said CAM [output from element 30].

- 3 -

13.   Because the same hardware and connections that are shown in Figures 1 and 2 of the '461 application are also shown in the Figures on Sheet 1/4 and Sheet 2/4, respectively, of Exhibit C, the invention disclosure dated June 27, 2001 supports claim 15 in the same manner that Figures 1 and 2 of the '461 application support claim 15.

14.   Claim 15 is also supported by the paragraph labeled Summary on page D– 1/8 and by the details of the invention disclosure as follows:

> inputting an Internet address to a plurality of hash circuits, each hash circuit being responsive to a different portion of said address [see paragraph labeled "Hardware Parallelism using Hashing Circuits and Matching in TCAM" on page D-3/8 and the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
> outputting a hash signal from each hash circuit [see first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
> using said hash signals to identify portions of a CAM [see first and third paragraphs under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
> inputting said address to said CAM [see first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
> comparing said address in only the identified portions of said CAM [see paragraph labeled "TCAM Matching" on page D-4/8, see paragraph labeled "TCAM Match Line Power Dissipation" on page D-5/8, and see third paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8]; and
> outputting port information in response to a match being found in said CAM [see the paragraph entitled "TCAM Output" on the bottom of page D-4/8 and top of page D-5/8].

15.   For each of the method limitations in claim 22, there is a corresponding hardware element shown in Figure 1 of the '461 application which provides that function. The correspondence is as follows:

> hashing different prefixes within an Internet address [see elements 24(1) – 24(n)];
> precharging certain portions of a CAM in response to said hashing [see enable lines output from element 26] ;
> comparing said Internet address in said precharged portions of the CAM [function of element 20]; and
> outputting information in response to a match being found in the CAM [see output line from element 20].

- 4 -

16. Because the same hardware and connections that are shown in Figure 1 of the '461 application are also shown in the Figure on Sheet 1/4 of Exhibit C, the invention disclosure of June 27, 2001 supports claim 22 in the same manner that Figure 1 of the '461 application supports claim 22.

17. Claim 22 is also supported by the details of the invention disclosure as follows:

hashing different prefixes within an Internet address [see paragraph labeled "Hardware Parallelism using Hashing Circuits and Matching in TCAM" on page D-3/8 and the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];

precharging certain portions of a CAM in response to said hashing [see paragraph labeled "TCAM Match Line Power Dissipation" on page D-5/8] ;

comparing said Internet address in said precharged portions of the CAM [see paragraph labeled "TCAM Matching" on page D-4/8 and the third paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8]; and

outputting information in response to a match being found in the CAM [see the paragraph entitled "TCAM Output" on the bottom of page D-4/8 and top of page D-5/8].

18. For each of the apparatus limitations in claim 28, there is a corresponding hardware element shown in Figure 1 of the '461 application. The correspondence is as follows:

a CAM for receiving a comparand word [see element 20];

a plurality of hash circuits connected in parallel, each for producing a hash signal in response to a portion of the comparand word [see elements 24(1) – 24(n)]; and

a circuit, responsive to said hash signals, for precharging portions of said CAM [see element 26].

19. Because the same hardware and connections that are shown in Figure 1 of the '461 application are also shown in the Figure on Sheet 1/4 of Exhibit C, the invention disclosure dated June 27, 2001 supports claim 28 in the same manner that Figure 1 of the '461 application supports claim 28.

20. Claim 28 is also supported by the details of the invention disclosure as follows:

a CAM for receiving a comparand word [see the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];

a plurality of hash circuits connected in parallel, each for producing a hash signal in response to a portion of the comparand word [see paragraph labeled "Hardware Parallelism using Hashing Circuits and Matching in TCAM" on page D-3/8 and the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8]; and

a circuit, responsive to said hash signals, for precharging portions of said CAM [see paragraphs 4 – 6 of the section entitled "3.2 Discussion of Circuit Figures"].

21.   For each of the apparatus limitations in claim 34, there is a corresponding hardware element shown in Figure 1 of the '461 application. The correspondence is as follows:

        a CAM [see element 20];
        a plurality of hash circuits each for producing a hash signal in response to a portion of a comparand word [see elements 24(1) – 24(n)];
        a plurality of memory devices responsive to said hash circuits [see elements 42(1) – 42(n) of Figure 2 of the '461 application];
        enable logic, responsive to said plurality of memory devices, for enabling portions of said CAM [see element 26]; and
        a delay circuit for inputting the comparand word to said CAM [see element 28].

22.   Because the same hardware and connections that are shown in Figures 1 and 2 of the '461 application are also shown in the Figures on Sheet 1/4 and Sheet 2/4, respectively, of Exhibit C, the invention disclosure dated June 27, 2001 supports claim 34 in the same manner that Figures 1 and 2 of the '461 application support claim 34.

23.   Claim 34 is also supported by the details of the invention disclosure as follows:

        a CAM [see the first paragraph under the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
        a plurality of hash circuits each for producing a hash signal in response to a portion of a comparand word [see paragraph labeled "Hardware Parallelism using Hashing Circuits and Matching in TCAM" on page D-3/8 and the first paragraph of the section entitled "3.2 Discussion of the Circuit Figures" on page D-5/8];
        a plurality of memory devices responsive to said hash circuits [see paragraphs 4 and 5 of the section entitled "3.2 Discussion of the Circuit Figures" on pages D-5/8 and D-6/8];
        enable logic, responsive to said plurality of memory devices, for enabling portions of said CAM [see paragraphs 4 – 6 of the section entitled "3.2 Discussion of Circuit Figures" on pages D-5/8 and 6/8]; and
        a delay circuit for inputting the comparand word to said CAM [see element 28].

24.   Each of the method limitations in claim 41 is supported by the details of the invention disclosure as follows:
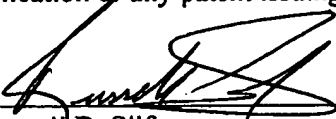
        transferring network addresses to a CAM based on an index to a hash table [see paragraph 3 of the section entitled "2. Summary" and the paragraph entitled "Loading hardware Before Routing" on page D-3/8];
        transferring port numbers to an output memory device responsive to the CAM [see the paragraph entitled "Loading Hardware Before Routing" on page D-3/8];
        modifying bit prefix values to obtain a ternary representation [see the paragraphs entitled "Hash Buckets" and "Pre-Designed Hash Tables" on page D-2/8];

- 6 -

calculating bank run length information [see the paragraph entitled "Bank-Indexed RAMs" on pages D-3/8 and D-4/8];

loading starting address and bank run length information into a plurality of memory devices [see paragraph 4 of the section entitled "3.2 Discussion of Circuit Figures" on page D-5/8 and 6/8]

See generally section 3.3 entitled "TCAM loading" on pages D-6/8 and 7/8.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of this application or any patent issuing thereon.
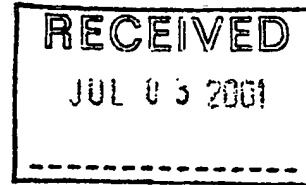
Russell D. Slifer .   2/6/07
                                        Date

INVENTION DISCLOSURE

1. INVENTOR(S):

2.   Keith R Slavin

3. DESCRIPTION:
  ● **Title:**

A Low Power Hash-Ternary Contents-Addressable Memory

  ● **Brief Description:**

A Ternary Contents-Addressable-Memory (TCAM) normally consumes a lot of power because of simultaneous activity on large numbers of match lines – one for each TCAM entry. This patent shows how this activity can be reduced in a guaranteed fashion for any random or systematically generated set of TCAM entries by using a form of hardware based hashing. An example is shown for 128-bit IPv6 internet address matching used in routers. The power savings increase as TCAMs get larger. Micron can use this patent to build TCAMs with unrivaled worst-case power consumption.

4. CONCEPTION & DOCUMENTATION OF INVENTION:
  ● **Date when first conceived:**

Redacted

  ● **To whom was the idea first described:**

Redacted

  ● **On what date:**

Redacted

  ● **Date of the first tangible record:**

Redacted

  ● **Type and location:**

Redacted

5. INFORMATION RELATED TO INVENTION:
  ● **Related invention disclosures:**

Redacted

Redacted

8. INVENTORS:

Name : Keith R Slavin

Home Address : 8474 SW Chevy PLace

City : Beaverton   State : OR   Zipcode : 97008

Citizenship : U.S.A

Company : Micron Technology, Inc.

Work Phone # : 503 643 1182   Mail Stop : 831

Dept Name : SJDC CA-ARCH-ADV DEV   Dept # : 402

Supervisor : Robert H. Mullis
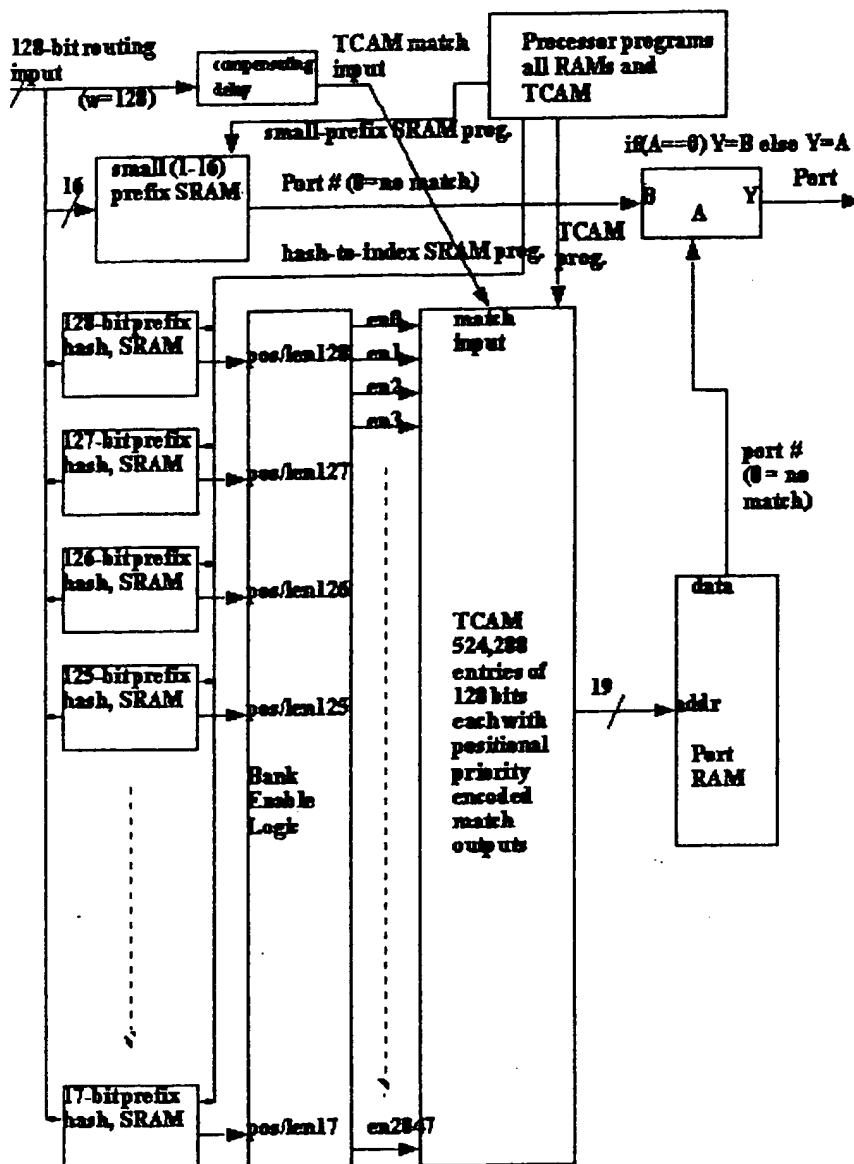
Signature : _Keith Slavin_   Date : 27th of June, 2001

27th June, 2001

Exhibit C-1/4

Figure 1. Block Diagram of Hash CAM System
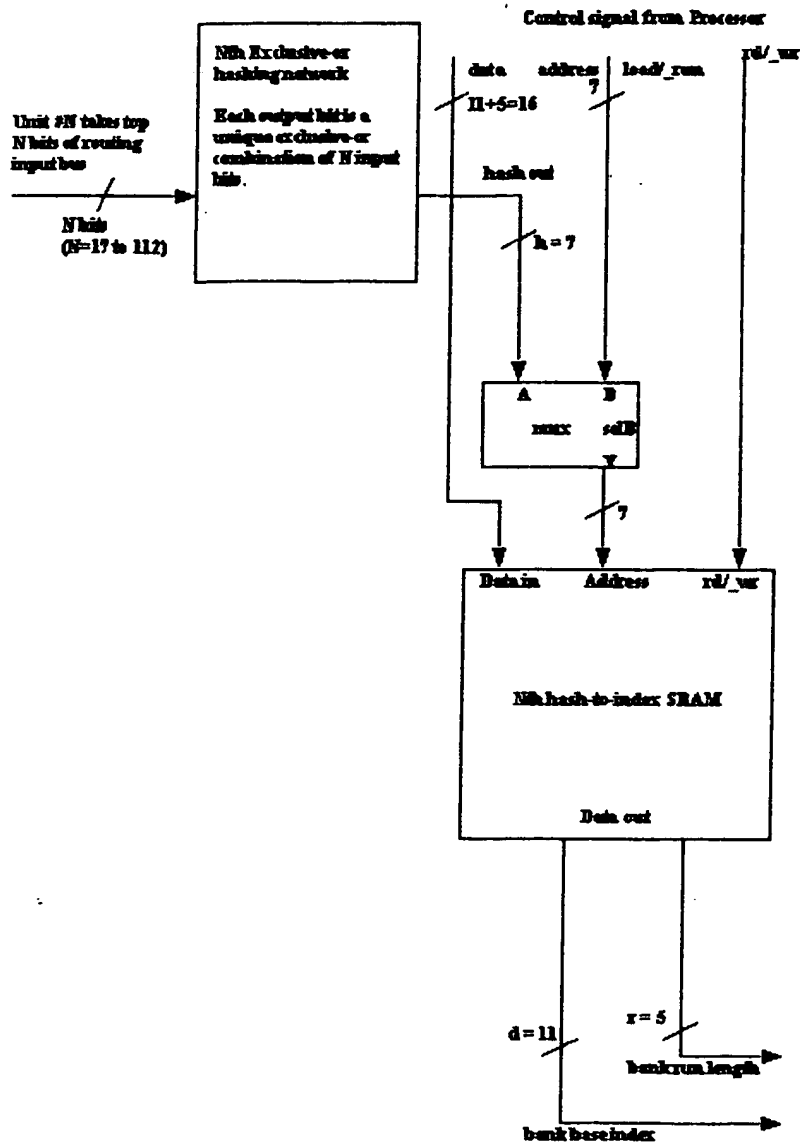
Exhibit C-2/4

Figure 2: Block Diagram of a Hash, SRAM (112 instances from Figure 1)

Read & Understood, Shawn 6/27/2001

Exhibit C-314

Figure 3: Bank Enable Logic Detail from Figure 1.

Exhibit C-414

N = 17 to 128 inclusive

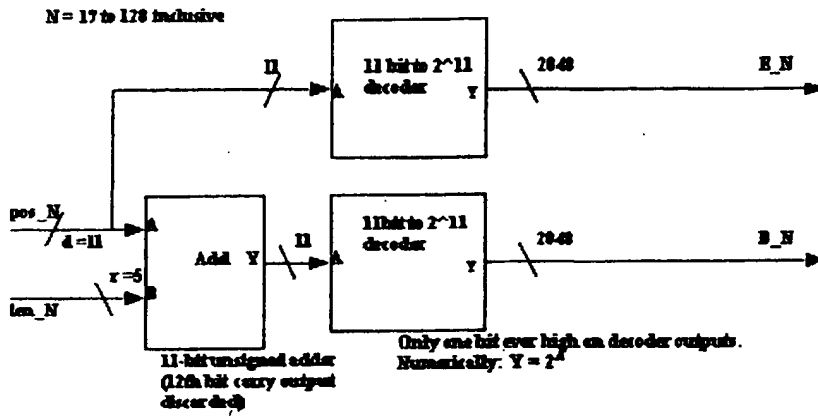**Figure 4: Bank Decoder Logic (112 instances from Figure 3)**

Redacted

2. Summary

Accordingly, the present invention provides a method of reducing the match line pre-charging activity for TCAM access while allowing operation at the highest possible TCAM speeds. The method involves sending the routing input to a TCAM for parallel comparisons, and also sending the routing input to a set of hash circuits, one for each legal prefix length. For each prefix length value, only that number of most significant bits are enabled from the routing input to the corresponding hashing circuit. Each hash circuit output value is therefore dependent on a different number of routing input bits. The number of bits on each hash circuit output can be optimized for the expected number of entries corresponding to the prefix length in typical usage. If the prefix length profile is unknown, each hash circuit output may have the same number of bits. Each hash circuit output then goes to the address of a corresponding RAM look-up which defines a region of the TCAM in which, according to the known hashing behavior, a match is guaranteed to be located - if such a match exists as a TCAM entry. Each RAM output is decoded and combined with the others to obtain an enable bus for banks of entries within the TCAM. Larger TCAM banks reduce complexity in the power-saving circuits, but enable more matches than the minimum required.

If more (including up to all) of the TCAM is enabled than is necessary, then the TCAM will still work normally, but useless entry matching will occur and TCAM power will rise.

A TCAM loading processor is responsible for sorting and writing the set of network address values into the TCAM in prefix length order, and for loading the RAM look-up tables associated with each prefix length.

A further refinement is to use an additional RAM for matching all short-prefix addresses in a routing table up to a pre-determined maximum length.

3. Description

Read & understood, s/lauren Lhn 6/27/2001

3.1 Understanding the Hash-TCAM System

Combining Hashes with TCAMs

The present invention is a fast, low-power, hash-TCAM system suitable for IP packet routing. The algorithm has a pre-processing and TCAM loading stage, followed by an operational mode which obtains port identifier values from an input value such as a forwarding IP address. From one perspective, the invention can be seen as a TCAM with additional power saving circuits. >From another perspective, it is a hashing system which uses a TCAM to find the match within each hash bucket.

Hash Buckets

A hash bucket is a term for a hash output value. Given that hashes usually accept many more input bits than the output bits they generate, hashes form a many-to-one function, so many possible inputs will map to a given hash bucket value.

Pre-Designed Hash Tables

For each possible prefix length, a hash function has previously been designed to work with it. Each hash function output is a function of all the bits within its prefix length. Each bit of a hash circuit output is a different function of the input bits than are the other bits in the function. The purpose of hashing in this case is to distribute any systematic or random set of input addresses reasonably evenly between its output bucket values, and to reduce the number of bits at the hash output compared to the hash input. This behavior is only guaranteed if knowledge of the hash functions is not used to generate the address set.

Initialization of Software Pre-Processing

At the start, a two-dimensional hash table is allocated - the major index is derived from the prefix length, and the minor index from the output of the hash function chosen for that prefix length. The hash table elements are actually references to linked lists. The hash table elements are all initialized to a null reference to show that each linked list is of zero length at this stage. The pre-processing then takes a given routing table, and performs a sort operation on each routing table entry using the hash functions and hash table.

Linked Lists

Linked lists are lists in which each member of a list contains a reference to the next member in the list. Linked lists have the advantage that list insertion (preferably into the start of the list) can be very fast, as only the links or references have to be changed, and no data has to be "bubbled up" to make room for new entries.

Creating Linked Lists from Routing Tables in Software

A routing table entry comprises an IP address, a prefix length, and an output port value. For each IP address, the associated prefix length is used to generate a mask to enable the top prefix length bits of the IP address through (and no other bits). The enabled bits are then applied to the hash function that was previously designed for that prefix length. The hash function output and prefix length are then used as

Karl & Understood, Shan—n 6/27/2001

the respective minor and major indices of the hash table. The value obtained from the hash table lookup is a reference to a linked list. The routing table entry, including the IP address, prefix length, and associated port value, is then inserted as an item into the referenced linked list.

Matching in Software Using the Hash Functions, Hash Table, and Linked Lists

To understand how the linked list data is used, a software search algorithm is described which uses the previously described set of hash functions, the 2-dimensional hash table, and the linked lists. Obtaining a port value from an IP address requires finding a match by searching over each possible prefix length. For each length, the corresponding masking and hash function is applied and the hash output and prefix length form the minor and major indices into the same hash table as was used in the pre-processing stage. The value from the hash table is a reference to the start of a linked list. The search for a match can then progress down the selected linked list, comparing the top prefix-length bits of the IP address input with the stored IP addresses from the routing table. If an exact match is found in the prefix bits, then the associated port value is returned. The port value associated with the longest prefix match is selected as the final result. This algorithm is fairly slow on a processor, although it can be sped up by working from the longest prefix downwards, and finishing immediately when a match is found.

Hardware Parallelism Using Hashing Circuits and Matching in TCAM

The above software algorithm can be implemented to run very fast in hardware, as i) all the hash functions (one per prefix length) can be performed in parallel using dedicated hash circuits, and ii) traversal of each linked list can be performed in parallel if the linked list data for matching is placed in a TCAM where all compares occur in parallel. In the software algorithm, the same address value is compared with each linked list entry, so the hardware search equivalent is achieved by placing all the linked lists in a TCAM, and enabling matching on those regions of the TCAM that hold the linked lists that would have been selected in the software solution.

Loading Hardware Before Routing

The hardware solution requires that the TCAM is loaded with the linked-list routing information before routing can occur. The linked list matching addresses are sequentially packet into the TCAM in order of prefix length (major index) and hash value (minor index). The format of the IP address and prefix length from each routing table entry is modified at some stage to represent the {0,1,X} format per bit used by the TCAM. The associated port number is not stored in the TCAM. Instead, the TCAM output is a priority encoded match address that is then used as an index into a RAM that contains the associated port values. The order of data transferred from within a linked list into the TCAM does not matter, as long as they are not stored interspersed with data from other linked lists.

Using TCAM Blocks to Reduce Hardware Complexity

The match enables of contiguous TCAM entries can be grouped into blocks, simplifying the block-enable circuits, while enabling more TCAM entries for matching than the true minimum. The resulting increase of power dissipated in the TCAM may be more than offset by the decrease in the size and dissipated power in the TCAM block-enable circuits.

Bank-Index RAMs

Kead ↾ understand, show him 6/27/200/

During the transfer into the TCAM, the positions of the start and end of each linked list may be quantized into block indices. The start block index and end block index for each linked list are programmed into bank-index RAMs - one RAM for each hash circuit output. The resulting hardware (see Figure 1) during packet routing, takes an IP address input and hashes it for each prefix length, using the same hash functions for each prefix as in the pre-loading stage. The output of each hash circuit is then sent to its bank-index RAM that in turn selects the banks required for matching in the TCAM, thereby saving power in the TCAM.

Using DRAM Technology to Implement Compact Dynamic TCAMs

DRAM technology can be modified to give dense TCAM storage with as few as 6 transistors per ternary storage cell. As DRAMs use a transistor and a capacitor per bit of storage, large TCAMs may be built with almost one quarter the storage density of DRAM.

Using an Additional RAM for Matching All Short Prefixes

For IPv6, 128-bit routing addresses are used. This represents far too many bits to be easily decoded with conventional RAM devices in such a manner that any prefix distribution in the routing tables can be handled, so a TCAM is ideal. For short prefix lengths, the matching functions performed by several hashing circuits and index RAMs can be replaced with a single, small, short-prefix RAM. Each extra bit dealt with by the short-prefix SRAM doubles its size, whereas each extra bit dealt with by hashing produces a linear circuit size increase. Therefore, an optimal number of short-prefix bits must exist for a given IC technology. In Figure 1, a short-prefix threshold of 16 bits is chosen as an example. If a match is found in the TCAM, it must have a longer prefix, so it over-rides prefix matches found by the short-prefix RAM. The short-prefix RAM is probably fairly small in practise, so it can be programmed quickly, and fast static-RAM (SRAM) can be used in high performance implementations.

TCAM and Associated Data RAM

The TCAM is programmed with data entries at addresses, similar to a RAM. The difference between RAM and TCAM is that when the TCAM is subsequently used for prefix matching, the TCAM input is usually simultaneously compared with all the TCAM entry data, and the address of a match is obtained - a sort of RAM-read-in-reverse. If more than one simultaneous TCAM match is possible, the highest priority match can then be priority encoded to an address to obtain data associated with the match using a conventional RAM. Associated data may include port values for routing purposes.

TCAM Matching

In compact TCAM designs, when a new routing input is received, the first step in the TCAM is to pre-charge all match lines to a known voltage, e.g. to a logic high or 1, and then release each match line drive to allow them to float near that voltage due to their inherent capacitance and the high impedance of other circuits on the match line. Each TCAM entry consists of a set of storage and comparison circuits corresponding to the routing input width. A short time after the match line is left floating, the TCAM comparison circuits are enabled. If even one of the comparison circuits stored value does not agree with the routing input, then the connected match line is pulled the other way, e.g. discharged to a logic low or 0.

TCAM Output

Read & understood, Sham——lu 6/27/20?/

Each match line is read with a sensing amplifier, and priority encoded according to its address in the TCAM. The priority encoded match output can then be used directly to select the associated port data.

TCAM Match Line Power Dissipation

Most of the time, few matches occur on a routing input to a TCAM, so most TCAM entries are miss-matched, and the corresponding match lines are then discharged from the pre-charged value. On the next access, the match lines are pre-charged again. This simultaneous activity on multiple match lines results in high power dissipation in the TCAM. To avoid this problem, match pre-charging, and possibly any other matching activity, can be disabled on those entries in which it is known that a match cannot exist.

Reliable TCAM Power Reduction

Any scheme for reducing power consumption has to be able to handle systematic patterns in routing tables, as IP address allocation schemes are unknown and may follow simple patterns in certain bits by region. A TCAM power saving scheme that ignores any routing input bits may deal with routing tables with routing table activity in these ignored bits, resulting in higher TCAM power. If the TCAM is a "worst-case" design, then the TCAM capacity may have to be reduced to allow for this - otherwise the TCAM may literally self-destruct!

3.2 Discussion of Circuit Figures

According to Figure 1, the routing input is sent to a TCAM for matching, and also sent to a small prefix RAM, and also to set of hash circuits, one for each prefix length. For example, if 128-bit IPv6 addressing is used, and small prefixes up to 16 are handled with an SRAM, then 128-16 = 112 prefixes use the TCAM, so 112 hash circuits are required. Each hash circuit deals with a different prefix length, so the first hash circuit processes the top 17 routing input bits, the next processes the top 18 bits, and so on to the last hash circuit which deals with the full 128 bits. The outputs of all the hash circuits are all shown as having $h = 7$ bits each in Figure 2. Each hash circuit output is also shown connected to the address of a small hash-to-index SRAM that maps each hash output value to define a region in the TCAM.

In a preferred embodiment, each hash circuit can be implemented using simple exclusive-or logic networks to generate each hash output bit from a randomly pre-determined set of hash inputs. Additionally, all the input bits to each hash function should contribute in some way to the overall output hash values.

For a TCAM with a large number of entries, specifying the exact position and size of a matching region requires much more logic circuits than less precise calculations. To reduce complexity in the power-saving logic, the TCAM is evenly divided into banks for match-enabling control purposes only. In the example in Figure 1, a 524,288 ($m = 19$-bit address) entry TCAM is divided into 2,048 banks, so each bank has 524,288 / 2048 = 256 entries. Each bank shares a single match enable input. In this way, the bank selection logic can be dramatically reduced in size, while still allowing significant power savings during TCAM matches.

In one embodiment, the start bank index and bank run-length are stored in each hash-to-index SRAM, as

Read & understood, ~~~~~~ 6/27/2001

the run-length is usually representable in fewer bits, and can be economically added to each base value to get the end position of each bank-enable region (see Figure 4).

In one embodiment, for each of the 112 prefix lengths of Figure 1, an inclusive start bank index and exclusive end bank index are both stored in each index SRAM and separately decoded (different to Figure 4).

According to Figure 3, the start bit positions from all 112 decoder circuits of Figure 4 are ored together to obtain a 2,048 bit bank enable bus. With 112 prefix lengths, up to 112 bits can be enabled at a time within this bus. Fewer bits may occur if information for more than one prefix start index lies within the same TCAM bank. The stop positions are similarly ored to obtain up to 112 disable bits in a 2,048 bit bank disable bus. A circuit then extents (replicates) all the enable bits upwards until each extension reaches a disable bit position. The bus now has 1's extending from each original start position up to and excluding the next stop position. The resulting 2,048-bit bus then controls the TCAM bank match enables to reduce power.

Most of the power-saving circuits consume very little power within and after the index decoding stages of Figure 3 (detailed in Figure 4), as most signals are inactive most of the time. The hash-to-index SRAMs are where most of the power-saving circuit itself consumes power.

3.3 TCAM loading

The TCAM loading algorithm is a vital part of the hash-TCAM system. The TCAM loading processor is first passed a routing table - a set of IP address values and a limited number of routing destinations. Routing destinations usually represent the IP addresses of known neighbors on the network. Each unique routing destination is mapped onto its own short port number before programming into the hash-TCAM system. The algorithm obtains the associated prefix lengths from the routing tables. Before loading the hardware, the software may check that all the network addresses for a given prefix length are different. The next step is to create and initialize a two-dimensional hash table in software memory. The table entries are set to a null reference to show that no linked lists are attached yet. The network address is also hashed in software, using the prefix value to select the same prefix bit mask and hash function as used for that prefix length in the hardware. The resulting hash output value and prefix length are then used as the minor index and major index respectively into the hash-table. Each array element points to the start of a linked list of values that represent all the prefixes with the same prefix length and the same hash value. The routing address, and associated port number is then inserted into the selected linked list. All network addresses are thereby sorted and inserted into appropriate linked lists in software.

The next step is to program the hardware in Figure 1, - i.e: the hash-to-index SRAMs, the short prefix SRAM, and hash-TCAM, and the port RAM. The processor sets a TCAM write address value to point to the start of the TCAM storage. The processor then systematically works through each prefix length, starting with the shortest prefix first.

For each prefix longer than the short-prefix SRAM can handle, the processor selects the hash table for that prefix. It then proceeds to work systematically through each hash index into the selected hash table, as follows:

For each hash table index, the processor reads the corresponding hash table value as a reference to the start of a linked list. It then goes down the linked list, transferring network addresses sequentially into

Read & understood, Shawn Lu 6/27/2011

the current TCAM address, and port numbers into the corresponding port RAM address. It then increments this address. The bit representation of IP addresses are modified with the prefix values to obtain the ternary {0,1,X} representations used for matching in the TCAM. The processor then calculates the inclusive first and exclusive last TCAM bank indices where the linked list has been written, and loads them as bank start and stop indices or alternatively as bank start index and bank run-length values into the hash-to-index SRAM for the current prefix. The index/run-length information comprise the SRAM data, and the output of the current hash function forms the SRAM address. Calculating bank indices from TCAM addresses is simple as long as TCAM bank lengths are chosen to represent powers of 2 in the TCAM address ranges.

The short-prefix SRAM can be separately loaded to handle all the shorter prefixes. In a preferred embodiment, the software hash tables used to load the TCAM are previously generated for all legal prefix lengths, and the short (16 or less in Figure 1) prefix hash tables are then used to expedite loading of this SRAM. Loading the short-prefix SRAM to deal with longest matching prefixes is well known in the art.

3.4 Fitting Routing Tables into a hash-TCAM system

The following circuit parameters are defined for a hash-CAM system:

m = address wires to write CAM entries

h = number of bits on hash circuit output (see Figure 2)

b = log2(number of entries in each TCAM bank)

d = m - b = number of decoder bits for bank selection (see Figure 4)

r = number of bits representing the maximum anticipated number of banks with same network address hash value from a routing table (see Figure 3 for bank run-length output, and the number of stages in 1's expansion in Section 3.4)

w = IP address length in bits (number of prefixes cannot be greater than w)

E = number of entries in the TCAM (cannot be greater than 2^m)

The probability that a given number of exactly n TCAM entries is associated with one hash value (in H possible hash values) can be obtained from the binomial theorem:

$$p[E, n, H] = \frac{E!}{(E-n)!n!}\left(\frac{1}{H}\right)^n\left(1 - \frac{1}{H}\right)^{E-n}$$

which gives the probability of n values in E entries in the TCAM being associated with the same hash value when H hash values are used. For each hash value, a maximum number N of TCAM entries is defined in which it is known that the matching prefix will be found, so n must lie in the range from 0 to N inclusive.

Read & understood, shown 6/27/200[?]

If an inclusive range of n from 0 to N is allowed for in hardware, then the probability of exceeding N is given by:

$$p(exceed) = 1 - \left(\sum_{n=0}^{N} p[E, n, H] \right)^{L}$$

L is the number of linked lists formed in the TCAM. The next step is to determine an upper value for r which is related directly to the maximum expected number of routing table sub-net addresses with the same prefix lengths that hash to the same value. In this case, fewer prefixes and the more entries gives the longest linked-lists and worst case fit. For one prefix, the expected number of bits required to encode the number of banks used per linked list is given by:

$$r = m - b - h$$

For the circuit of Figure 1, m = 19, b = 8, h = 7, which gives r = 4, or a maximum of $2^r - 1 = 15$ banks to hold each linked list. This average size is obviously insufficient when the TCAM is full ($E = 2^m$), so r is increased to 5 and analyzed. The maximum number of entries per linked-list supported in this case is N $= (2^r - 1)2^b = 7,936$. If the TCAM is full, then the chance that N or more entries are needed for a random hash distribution is:

$$p(exceed) < 10^{-618}$$

which is an astronomically low probability as long as hashing behaves in the expected manner. Two or more prefixes reduce the probability of no fit even further.

Redacted

Read & understood, Shane Lu 6/27/2001